Algorithm: Training ASR model for Cypriot dialect

Algorithm 1: Training ASR model for Cypriot dialect

```
Input: Raw dataset $D$ with fields {audio, text}
Output: Fine-tuned Wav2Vec2-CTC model $\theta^*$, evaluation metric (WER)
procedure TRAIN_ASR_ CYPRIOT(D)
    // 1) Load data
    $ds \leftarrow$ load_dataset(...);
    $D_{train} \leftarrow ds$["train"];
    $D_{eval} \leftarrow ds$["validation"];
    // 2) Resample audio to 16 kHz
    foreach $s \in \{D_{train} \cup D_{eval}\}$ do
        if $s.audio.sampling\_rate \neq 16000$ then
            $s.audio.array \leftarrow$ librosa.resample($s.audio.array, s.audio.sr$, 16000);
            $s.audio.sr \leftarrow$ 16000;
    // 3) Load processor & model (Greek Wav2Vec2)
    processor ← AutoProcessor.from_pretrained(...);
    model ← AutoModelForCTC.from_pretrained(...);
    freeze(model.feature_extractor);
    // 4) Feature extraction + label tokenization
    Define function PREPARE_BATCH(B):;
        $X \leftarrow [b.audio.array \mid b \in B]$;
        $Y \leftarrow [b.text \mid b \in B]$;
        inputs ← processor($X$, sampling_rate=16000, padding, truncation, max_length=16000);
        $y_{lower} \leftarrow$ lowercase($Y$);
        labels ← processor.tokenizer($y_{lower}$, padding, truncation, max_length=512);
        $L \leftarrow$ tensor(labels.input_ids);
        $L[L == 54] \leftarrow -100$                    // ignore padding token;
        return $\{\text{inputs}, \text{labels} : L\}$;
    $TrainDict \leftarrow$ PREPARE_BATCH($D_{train}$);
    $EvalDict \leftarrow$ PREPARE_BATCH($D_{eval}$);
    $TrainHF \leftarrow$ Dataset.from_dict($TrainDict$);
    $EvalHF \leftarrow$ Dataset.from_dict($EvalDict$);
    // 5) Data collator
    Define DataCollator(processor): dynamic padding, ignore pad in loss;
    // 6) Metric (WER)
    wer_metric ← evaluate.load("wer");
    Define COMPUTE_METRICS(preds): decode predictions & compute WER;
    // 7) Training configuration
    args ← TrainingArguments();
    // 8) Trainer
    trainer ← Trainer();
    // 9) Optimize
    $\theta^* \leftarrow$ trainer.train();
    return $\theta^*, trainer.evaluate()$;
```